Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ

По лабораторной работе №8

По курсу: «Функциональное и логическое программирование»

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

Преподаватели: Толпинская Н. Б.,
Строганов Ю. В.

Москва

2020

Задание 1. Написать функцию, которая по своему списку-аргументу `lst` определяет, является ли он палиндромом (то есть равны ли `lst` и `(reverse lst)`).

```
(defun is_palindrom (lst)
  (and (equal (null lst) nil)
       (equal (reverse lst) lst)))
```

Задание 2. Написать предикат `set-equal`, который возвращает `t`, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.

```
(defun set-equal1 (a b)
  (and (null (set-difference a b))
       (null (set-difference b a))))

(defun set-equal2 (a b)
  (not (set-exclusive-or a b)))
```

Задание 3. Напишите функцию `swap-first-last`, которая переставляет в списке-аргументе первый и последний элементы.

```
(defun swap-first-last (x)
  (cond ((null (cdr x)) x)
        (t (append (last x) (cdr (butlast x)) (list (car x))))))
```

Задание 4. Напишите функцию `swap-two-elements`, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

```
(defun swap-two-elements1 (x i1 i2)
  (cond ((null (rotatef (nth i1 x) (nth i2 x))) x)))

(defun swap-two-elements2 (x i1 i2)
  (let ((i (min i1 i2)) (j (max i1 i2)))
       (append
         (subseq x 0 i)
         (list (nth j x))
         (subseq x (+ i 1) j)
         (list (nth i x))
         (subseq x (+ j 1)))))

(defun swap-two-elements3 (x i1 i2)
  (let ((i (min i1 i2)) (j (+ (max i1 i2) 1)))
    (append
      (subseq x 0 i)
      (swap-first-last (subseq x i j))
      (subseq x j))))
```

Задание 5. Напишите две функции, `swap-to-left` и `swap-to-right`, которые производят круговую перестановку в списке-аргументе влево и вправо, соответственно.

```
(defun swap-to-left (lst &optional (k 1))
  (let ((k (rem k (length lst))))
       (append (subseq lst k) (subseq lst 0 k))))

(defun swap-to-right (lst &optional (k 1))
  (setf len (length lst))
  (let ((k (- len (rem k len))))
       (append (subseq lst k) (subseq lst 0 k))))
```

Задание 6. Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда а) все элементы списка — числа, б) элементы списка — любые объекты.

```
(defun mul_n (lst n)
  (mapcar (lambda (x) (* n x)) lst))

(defun mul_all (lst n)
  (mapcar
    (lambda (x)
      (cond
        ((numberp x) (* n x))
        ((listp x) (mul_all x n))
        (t x)))
    lst))
```

Задание 7. Напишите функцию, `select-between`, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел).

```
(defun select-between (lst a b)
  (let ((a (min a b)) (b (max a b)))
       (sort
         (remove-if-not (lambda (x)
           (and (<= a x) (<= x b))) lst)
         #'<)))
```